MVC šablon

MVC (skraćeno od Model-View-Controller) radni okvir (framework) ili šablon predstavlja način razvijanja aplikacije koji deli aplikaciju na tri glavna dela: model, prikaz i kontroler. Svaka od ovih komponenti je kreirana radi rukovanja specifičnim aspektom aplikacije. MVC se tradicionalno koristio najviše za desktop grafičke aplikacije, a kasnije je ovaj model razvijanja i razmišljanja prebačen i na web aplikacije, kao i na mobilne aplikacije.

MVC arhitektura se prvi put pominje 1979. godine, kada je norveški naučnik Trygve Reenskaug predstavio ovu svoju zamisao, dok se prvi put kao koncept koristi u programskom jeziku Smalltalk 1987. godine.

Povezanost

Pored podele same aplikacije u tri različite komponente, model-prikaz-kontroler šablon definiše i njihovu međusobnu povezanost. Model je zaslužan za upravljanje podacima aplikacije. Tačnije, on prima korisničke upite od kontrolera. Prikaz je zapravo izgled same aplikacije, dok kontroler reaguje na akcije korisnika i vrši interakciju nad podacima modela. Kontroler prima korisničku akciju, opciono je validira i šalje modelu, pa se tako može reći da predstavlja spregu između modela i prikaza.

Komponente

Model predstavlja centralnu komponentu MVC šablona. Odgovara svoj logici koja se tiče podataka sa kojima korisnik radi. Direktno upravlja tim podacima, logikom i uslovima/pravilima date aplikacije.

Prikaz može biti bilo koja reperezentacija podataka, kao što je, na primer, dijagram, grafik ili tabela. Takođe je moguće da se isti podaci prikazuju u više različitih prikaza.

Treća komponenta, kontroler, prima akcije korisnika i pretvara ih u komande ili za model ili za prikaz. Služi kao sprega između modela i prikaza koja obrađuje svu logiku i nadolazeće zahteve korisnika, upravlja podacima koristeći model i koristi prikaz za finalni prikaz podataka.

Model

Model predstavlja komponentu koja skladišti podatke i logiku vezanu za njih. Na primer, predstavlja podatke koji se prenose između kontrolera. Kontroler će dopremiti informacije o korisniku iz baze kako bi model mogao da radi na tim podacima i pošalje ih nazad u bazu ili prikaže korisniku.

Prikaz

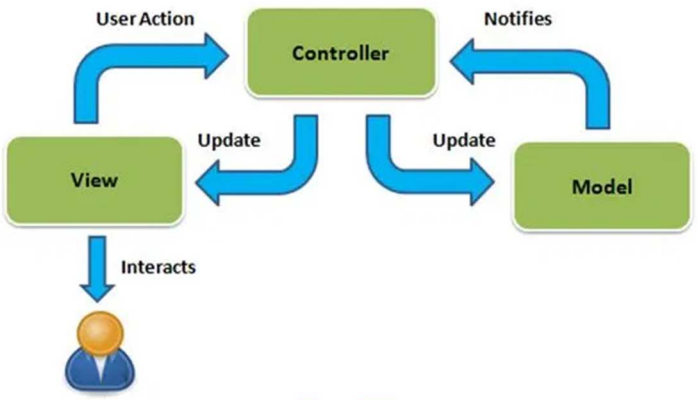
Prikaz je deo aplikacije koji predstavlja prikaz podataka (vizuelnu reprezentaciju). Prikaz se kreira pomoću podataka prikupljenih iz modela. Prikaz šalje zahtev modelu za određenim

Copyright © Link group

informacijama kako bi mogao da ih prikaže krajnjem korisniku. Takođe, prikaz prezentuje podatke iz dijagrama, tabela, grafikona, padajućih menija, boksova sa tekstom i slično.

Kontroler

Kontroler je deo aplikacije koji je zadužen za korisničku interakciju. Kontroler raspoznaje ulaze korisnika u vidu korišćenja miša i tastature i o tome informiše model i prikaz komponente o eventualnim zadatim promenama. Na primer, kontroler može poslati komandu modelu da ažurira podatke o određenom korisniku u bazi, ili snimi otvoreni dokument. Takođe, kontroler šalje komandu prikazu da promeni izgled ekrana koji korisnik vidi (na primer, pomeranje točkića miša bi pomeralo sadržaj dokumenta nagore ili nadole).



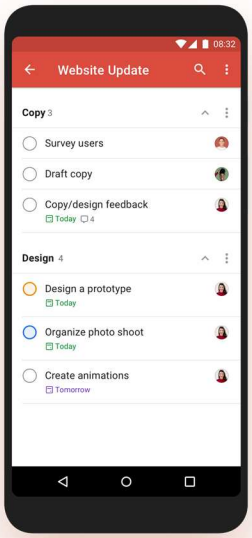
Slika 5.1. Grafički prikaz MVC šablona[1]

Primena MVC šablona

MVC šablon nam pomaže prilikom planiranja aplikacije, najviše jer nam prikazuje kako naše ideje mogu biti prebačene u kod. Recimo da kreiramo To-Do aplikaciju. Ova aplikacija bi omogućavala korisnicima da kreiraju zadatke i organizuju ih po tematskim listama. Model bi u ovom slučaju definisao šta je zadatak i da je lista zapravo kolekcija zadataka. Kod prikaza bi definisao kako vizuelno zadaci i liste izgledaju. Na primer, imena lista bi imala veći font nego sami zadaci, podebljan font itd. Na kraju, kontroler bi definisao način na koji korisnik dodaje zadatak ili ga obeležava kao završen. Sam kontroler povezuje dugme na prikazu. Jedan takav primer možemo videti na slici ispod, gde su tematske celine Copy i Design. Liste zadataka unutar tih celina su modeli (Survery users, Draft copy, Design a prototype itd.).

[1] https://thedotnetguide.com/mvc-design-pattern/

Copyright © Link group



Slika 5.2. Primer To-Do aplikacije koji je zasnovan na MVC šablonu[2]

Pitanje

Kontroler je:

- komponenta koja skladišti podatke i logiku vezanu za njih
- deo aplikacije koji predstavlja prikaz podatka
- **deo aplikacije zadužen za korisničku interakciju**

Objašnjenje:

Tačan odgovor je da je kontroler deo aplikacije zadužen za korisničku interakciju.

Prednosti MVC modela:

- olakšano dopunjavanje koda i dodavanje novih mogućnosti;
- olakšana podrška za nove tipove klijenata;
- razvoj različitih komponenti se može odvijati paralelno;
- pogodan je za velike web aplikacije iza kojih stoji dosta ljudi;
- omogućava logično grupisanje srodnih akcija.

Mane korišćenja MVC-ja:

- teže tumačenje, menjanje i testiranje koda;
- navigacija kroz kodnu bazu može biti kompleksan proces;
- neefektivnost podataka;

[2] https://todoist.com/

Copyright © Link group

- MVC zahteva poznavanje više različitih programerskih tehnologija;
- pošto je potrebno poznavanje više različitih programerskih tehnologija, više programera je potrebno za rad na MVC aplikaciji.

Primer MVC šablona u Pythonu:

Primer je baziran na ideji „biblioteke" – softvera koji je namenjen čuvanju knjiga. Deklarisaćemo sve tri komponente koristeći programski jezik Python:

Model.py

```
class Book:
    def __init__(self, title = None, year = None):
        self.title = title
        self.year = year

    #returns all people inside db.txt as list of Person objects
    def getAll(self):
        books = [{'title':'The Picture of Dorian Gray','year':'1890'},
            {'title':'Pride and Prejudice','year':'1813'},
            {"title":'The Adventures of Tom Sawyer ', 'year':'1875'}]
        return books
```

U našem primeru biblioteke, model će nam predstavljati fajl sa klasom Book, koji zapravo predstavlja knjigu. Radi pojednostavljenja, naša baza je zapravo lista rečnika sa dva ključ–vrednost para: ime knjige i godina izdavanja, koji se u program vraćaju pozivom getAll() metode klase Book.

View.py:

```
def showAllView(list):
    print ('In our db we have {} books. Here they
are:'.format(len(list)))
    for item in list: print ("\t- {}".format(item['title']))

def startView():
    print ('MVC - example start:')

def endView():
    print ('End.')
```

U našem primeru biblioteke, kod prikaza nam zapravo služi za komunikaciju sa korisnikom. Naime, prikaz će u ovom slučaju na ekranu prikazivati informacije koje kontroler prosledi. Funkcije startView() i endView() su tu za informaciju o pokretanju, odnosno zaustavljanju programa, dok nam funkcija showAllView() zapravo ispisuje na ekran čitavu bazu knjiga.

Controller.py:

```
from Model import Book
import View
def showAll():
```

Copyright © Link group

```
    #gets list of all Book objects
    b = Book()
    books_in_db = b.getAll()

    #calls view
    return View.showAllView(books_in_db)

def start():
    View.startView()
    showAll()
    View.endView()

if __name__ == "__main__":
    #running controller function
    start()
```

Kompletan kod primera možete pokrenuti i ujedno testirati u okviru razvojnog okruženja.

Radno okruženje

Model.py:

```
class Book(object):
    def __init__(self, title = None, year = None):
        self.title = title
        self.year = year

    #returns all people inside db.txt as list of Person objects
    def getAll(self):
        books = [{'title':'The Picture of Dorian Gray','year':'1890'},
            {'title':'Pride and Prejudice','year':'1813'},
            {"title":'The Adventures of Tom Sawyer ', 'year':'1875'}]
        return books
```

View.py:

```
def showAllView(list):
    print ('In our db we have {} books. Here they
are:'.format(len(list)))
    for item in list: print ("\t- {}".format(item['title']))

def startView():
    print ('MVC - example start:')

def endView():
    print ('End.')
```

Controller.py:

```
from Model import Book
import View
```

Copyright © Link group

```
def showAll():
   #gets list of all Book objects
   b = Book()
   books_in_db = b.getAll()

   #calls view
   return View.showAllView(books_in_db)

def start():
    View.startView()
    showAll()
    View.endView()

if __name__ == "__main__":
   #running controller function
   start()
```

U našem primeru softvera za biblioteku, kontroler je zadužen za povezivanje podataka sa onim što korisnik vidi na ekranu (modela sa prikazom). Konkretno u ovom fajlu, prvo uvozimo kod modela i prikaza (from Model import Book, import View). Pri pokretanju fajla Controller.py, izvršava se funkcija start(). Ispis nakon završetka ovog programa izgleda ovako:

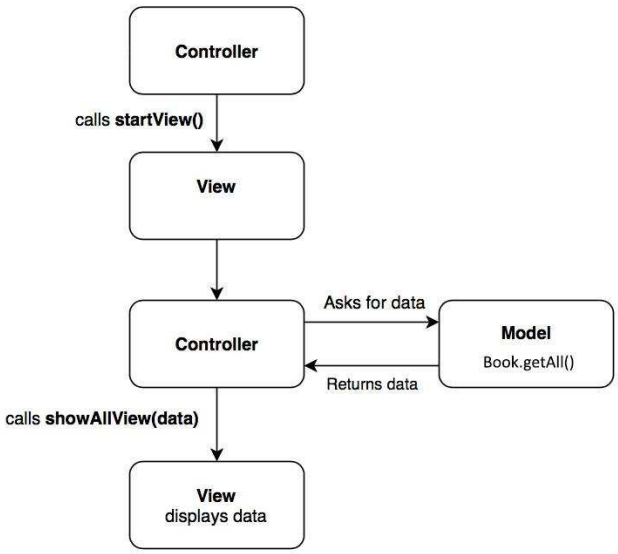
Ispis MVC primera:

```
MVC - example start:
In our db we have 3 books. Here they are:
        -The Picture of Dorian Gray
        -Pride and Prejudice
        -The Adventures of Tom Sawyer
End.
```

Detaljnijom analizom koda možemo videti ono što zapravo i piše u funkciji start() kontrolera – a to je da se prvo izvršava funkcija.

Dijagram našeg primera izgleda ovako:

Copyright © Link group



Slika 5.3. Dijagram MVC primera – biblioteka

Rezime

- MVC (skraćeno od Model-View-Controller) radni okvir (framework) predstavlja način razvijanja aplikacije koji deli aplikaciju na tri glavna dela: model, prikaz i kontroler.
- MVC arhitektura se prvi put pominje 1979. godine, kada je Trygve Reenskaug predstavio ovu svoju zamisao, dok se prvi put kao koncept koristi u programskom jeziku Smalltalk 1987. godine.
- Model predstavlja komponentu koja skladišti podatke i logiku vezanu za njih.
- Prikaz je deo aplikacije koji predstavlja prikaz podataka (vizuelnu reprezentaciju).
- Kontroler je deo aplikacije koji je zadužen za korisničku interakciju.

Copyright © Link group